



**HARNESSING AI FOR FRONT-END DEVELOPMENT: AN EVALUATION OF
DESIGN-TO-CODE TOOLS**

Lappeenranta–Lahti University of Technology LUT

Bachelor's Programme in Information Technology, Bachelor's thesis

2024

Juhani Manninen

Examiner: Associate Professor Jussi Kasurinen

ABSTRACT

Lappeenranta–Lahti University of Technology LUT

LUT School of Engineering Sciences

Software Engineering

Juhani Manninen

Harnessing AI for Front-End Development: An Evaluation of Design-to-Code Tools

Bachelor's thesis

2024

27 pages, 8 figures, and 2 tables

Examiner: Associate Professor Jussi Kasurinen

Keywords: AI, artificial intelligence, front-end development, Figma, design-to-code, AI-generated code

The recent development in AI technology has revolutionized various industries, including web development. Innovative design-to-code AI tools offer promising ways to streamline front-end development by translating design mockups into functional code.

The goal of this bachelor's thesis is to examine five of these tools to find out what their capabilities and limitations are, and if they could be used in real-world applications. Tools were tested on two website mockups created in the design program Figma, focusing on the accuracy of the results, available features, limitations, and user experience. Based on the results of this study, Locofy and Anima could be used to streamline the early stages of front-end development by creating prototypes, but programming is still required for pixel-perfect designs and adding functionality to the page. The greatest challenge tested tools faced was adapting to different screen sizes and implementing advanced features like animations and working links.

TIIVISTELMÄ

Lappeenrannan–Lahden teknillinen yliopisto LUT

LUTin insinööritieteiden tiedekunta

Tietotekniikka

Juhani Manninen

Tekoälytyökalujen hyödyntäminen front-end kehityksessä ja niiden arviointi

Tietotekniikan kandidaatintyö

2024

27 sivua, 8 kuvaa ja 2 taulukkoa

Tarkastaja: Tutkijatohtori Jussi Kasurinen

Avainsanat: AI, tekoäly, front-end kehitys, Figma, design-to-code, tekoälyn tuottama koodi

Viimeaikainen kehitys tekoälyteknologiassa on mullistanut eri teollisuudenaloja, mukaan lukien verkkokehityksen. Innovatiiviset design-to-code tekoälytyökalut tarjoavat lupaavia mahdollisuuksia front-end-kehityksen nopeuttamiseksi, muuntamalla verkkosivumallit toimivaksi koodiksi.

Tämän kandidaatintutkielman tavoitteena on tutkia viittä näistä työkaluista, selvittää mihin ne pystyvät ja mihin ei. Siten saadaan selville voisiko niitä hyödyntää tuotannossa. Työkaluja testattiin kahdella Figmalla luotuja verkkosivumalleja, keskittyen tulosten tarkkuuteen, saatavilla oleviin ominaisuuksiin, rajoituksiin ja käyttökokemukseen. Tulosten perusteella Locofy ja Anima voisivat soveltua varhaisten prototyyppien tekemiseen, mutta ohjelmointi on edelleen tarpeen virheiden korjaamiseen ja toiminnallisuuksien lisäämisen takia. Suurin haaste testatuille työkaluille oli sopeutuminen eri näyttökokoihin ja edistyneiden ominaisuuksien, kuten animaatioiden ja toimivien linkkien, toteuttaminen.

SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
TS	TypeScript

Table of contents

Abstract

Symbols and abbreviations

1	Introduction	6
1.1	Goals and Limitations	6
1.2	Thesis Structure	7
2	Background.....	8
2.1	Terminology.....	8
2.1	Related Research.....	10
3	Research Method	12
3.1	Practical Analysis	12
4	Results	16
4.1	Anima.....	16
4.1	Builder.io	17
4.1	Clapy.....	19
4.2	Locofy.....	20
4.1	TeleportHQ	21
4.1	Summary of the AI Tools	23
5	Discussion.....	24
5.1	Key Findings and Challenges	24
5.2	Design-to-Code Tool Use Cases.....	24
5.1	Aligning Study Results with Existing Literature	25
6	Conclusions	26
	References.....	27

1 Introduction

The web development landscape is constantly changing, with ever-increasing demands for high-quality websites that need to be built more efficiently. Traditional development workflows, which involve hand-coding websites from scratch, can be time-consuming and prone to errors. This is where design-to-code AI tools are emerging as a potential game changer. They promise to revolutionize web development by automatically generating production-ready code from detailed design mockups created in programs like Figma. While existing research has explored various approaches to convert web designs into components and entire websites, a comprehensive comparison of different AI-powered tools hasn't been studied before.

1.1 Goals and Limitations

This research investigates the potential of design-to-code AI tools by evaluating their capabilities and limitations. By analyzing how effectively these tools translate design elements into code, we can gain valuable insights into their real-world usability for web development. This research aims to answer two key questions:

- What are the capabilities and limitations of currently available design-to-code AI tools?
- How effective are these tools in expediting the web development process?

By answering these questions, this thesis aims to contribute to a better understanding of this emerging technology and its potential impact on the future of web development.

The evaluation will focus on the output and functionality generated by the AI tools, comparing them to the original design mockups. The inner workings of the AI algorithms and the quality of the generated code itself are outside the scope of this research.

1.2 Thesis Structure

Chapter 2: In this chapter, a review of existing research on AI code generation in web development is conducted. Additionally, key terminologies relevant to the field are defined to establish a solid foundation for the following chapters.

Chapter 3: This elaborates on the research methodology employed in this study. It outlines the testing procedures used for evaluating the effectiveness of design-to-code AI tools.

Chapter 4: This chapter presents the findings from the testing and evaluation of the tools. The results are thoroughly analyzed to provide insights into the performance and capabilities of these tools.

Chapter 5: This chapter delves into the implications and interpretations derived from the research findings. It explores the significance of the results and their potential impact on the field of AI code generation in web development.

Chapter 6: The final chapter summarizes the key takeaways and contributions of the research. It encapsulates the main findings, implications, and recommendations.

2 Background

This chapter begins by defining key terms related to front-end development and how they work together to create the visual and interactive elements of a website. Then the chapter delves into related research on automated code generation.

2.1 Terminology

Front-end development refers to the creation of the graphical user interface for the website. It focuses on how a website looks and how users interact with it. Several programming languages are used to create them, and each serves a distinct purpose in constructing web applications. HTML is often referred to as a skeleton of a webpage. It defines the structure and content using tags that specify headings, paragraphs, images, lists, etc. CSS is responsible for the visual presentation of the webpage. It controls the layout, styling, and overall aesthetics, essentially clothing the HTML skeleton. JS enhances web pages by introducing interactivity and functionality through animations, user input management, form validation, and a range of other features, bringing the website to life. While these three languages are the fundamental building blocks of front-end development, front-end frameworks and libraries are often used to develop more complex and efficient web applications. Examples include React, Angular, and Vue.js. These tools improve website scalability, performance, and development processes. A wide variety of features in AI tools makes them more useful and accessible for developers. [1], [2]

The purpose of design-to-code AI tools is to automatically convert mockup designs into functional code. They can interpret design elements and structures to generate corresponding HTML, CSS, and JS while supporting multiple frameworks and libraries. This process can significantly speed up the development process because the developer doesn't have to code the designs line-by-line. This provides additional time for refining interactions and logic on the website, enhancing the quality of the produced website. [3], [4] This thesis uses AI tools, tools, and design-to-code tools asynchronously, meaning the same thing.

Artificial Intelligence (AI) is a field that involves machines, particularly computer systems, simulating human intelligence processes. This encompasses a wide range of applications such as expert systems, natural language processing, speech recognition, and machine vision. Expert systems are designed to replicate the decision-making abilities of human experts in specific domains by utilizing knowledge bases and inference engines. Natural Language Processing (NLP) enables computers to understand, interpret, and generate human language, facilitating applications like chatbots, language translation, sentiment analysis, and text summarization. Speech recognition technology allows AI systems to convert spoken language into text, powering virtual assistants like Google Assistant, Siri, and Alexa, as well as transcription services and voice-controlled applications. Machine vision capabilities enable machines to interpret visual information from the environment for tasks such as image and video analysis, object recognition, and image processing. [5]

Figma is a popular design tool for many applications, including web design. It enables quick prototyping and easy collaboration between team members and stakeholders. Additionally, it supports many plugins, including the design-to-code tools this thesis is studying. Figure 1 showcases Figma's design environment.

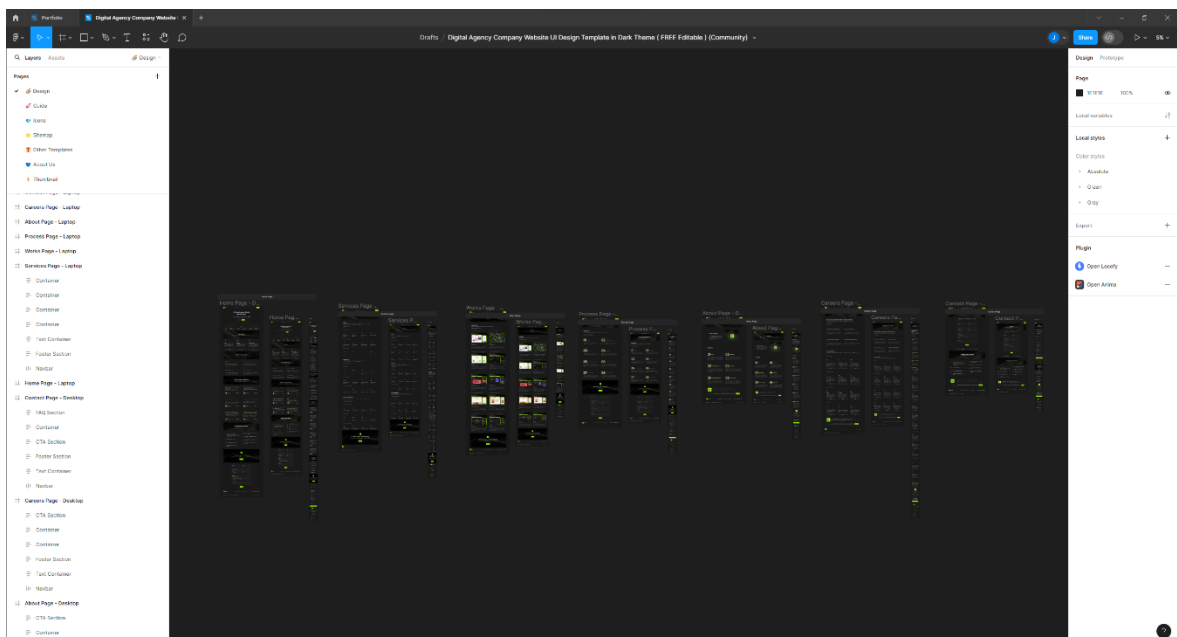


Figure 1. Company website design in Figma.

2.1 Related Research

Research conducted by university students discusses developing a technique to automatically generate HTML code from hand-drawn mock-up images using machine learning techniques like convolutional neural networks, deep learning, and object recognition. By applying these techniques components can be identified, classified, and converted to HTML code using an HTML builder algorithm. A dataset of mock-up images containing different elements was created and the CNN model was trained on it to accurately detect objects. [6]

Research written by Adelola and Oladokun explored both the advantages and challenges associated with automated code generation using OpenAI API, providing insights into emerging trends. It detailed the benefits of automated code generation, explaining how it can speed up development processes, improve code quality and consistency, reduce human errors, and simplify maintenance tasks. The study also brought out limitations of AI-generated code, including issues with accuracy and reliability, handling complex design requirements, adapting to specific project needs, considering ethical implications, and emphasizing the importance of human oversight. Furthermore, the research identified practical applications for automated code generation in real-world scenarios, such as supporting rapid prototyping, seamlessly integrating with UI/UX design tools, powering E-commerce platforms, and enhancing content management systems. [7]

A study conducted at the University of British Columbia created a program that automatically generates reusable web components from user interface mock-ups, reducing manual work for developers. It analyses mock-ups using visual cues and unsupervised learning, achieving an average of 94% precision and 75% recall in agreement with expert developers. Additionally, the generated components offer an average of 22% code reusability, demonstrating the potential of the program to streamline web development workflows. [8]

Master's thesis by Albin Frick aimed to address the challenge designers and developers face when creating reusable components for multiple projects with differing designs. It presented a prototype that automatically generates web components from Figma design files. Interviews provided insight into roles' workflows to ensure the prototype was useful and usability testing evaluated the prototype's ease of use. The results demonstrated the prototype's ability to convert Figma designs to functioning web components, though further development was still required. [9]

3 Research Method

This thesis is an exploratory case study that focuses on qualitative results. The case study research method involves an in-depth examination of one or a few cases to gain a comprehensive understanding of a phenomenon within its real-life setting. This methodology is not only valuable for evaluating development methods but also for observing, explaining, and exploring various phenomena. Case studies provide a systematic approach to looking at events, collecting data, analyzing information, and reporting results. They can be classified based on their purpose as descriptive, explanatory, exploratory, or evaluatory. Exploratory studies aim to gain new insights and generate ideas for new research. [10], [11]

3.1 Practical Analysis

To ensure a comprehensive evaluation, the AI tools were tested on two contrasting design files (Figures 2 & 3). The first design represents a software engineer's portfolio, created by the author with a limited background in design principles and Figma. The second design is a complex multi-page company website crafted by a professional designer with three different screen sizes. This diversity in design complexity allows analysis of the tools' effectiveness in handling both basic and detailed layouts. Due to the maximum page limitations of the AI tools, only desktop designs were converted when testing the company website page. This also provided more information on how the tools can translate desktop design into mobile.

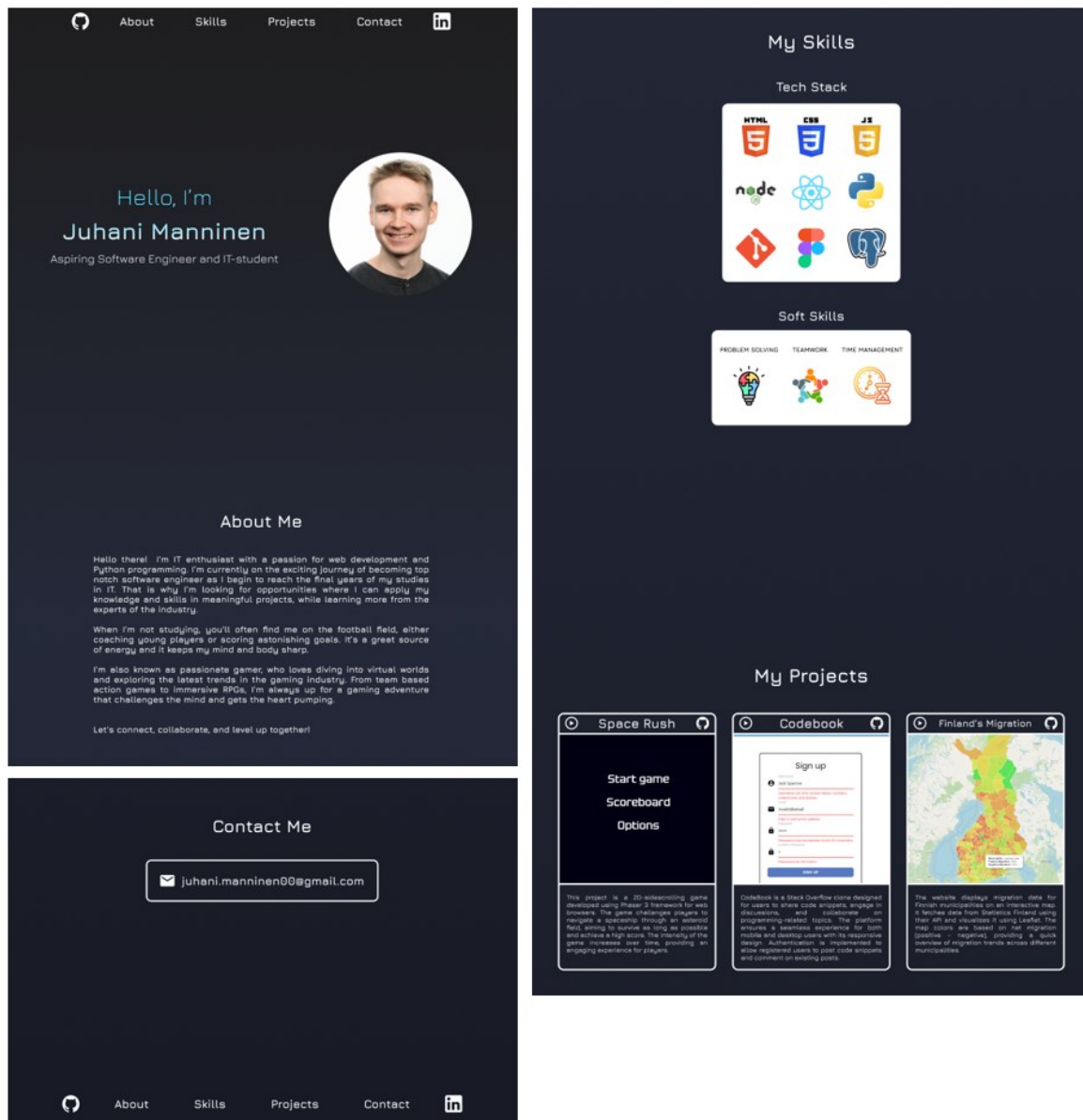


Figure 2. Tested portfolio design made by the author.

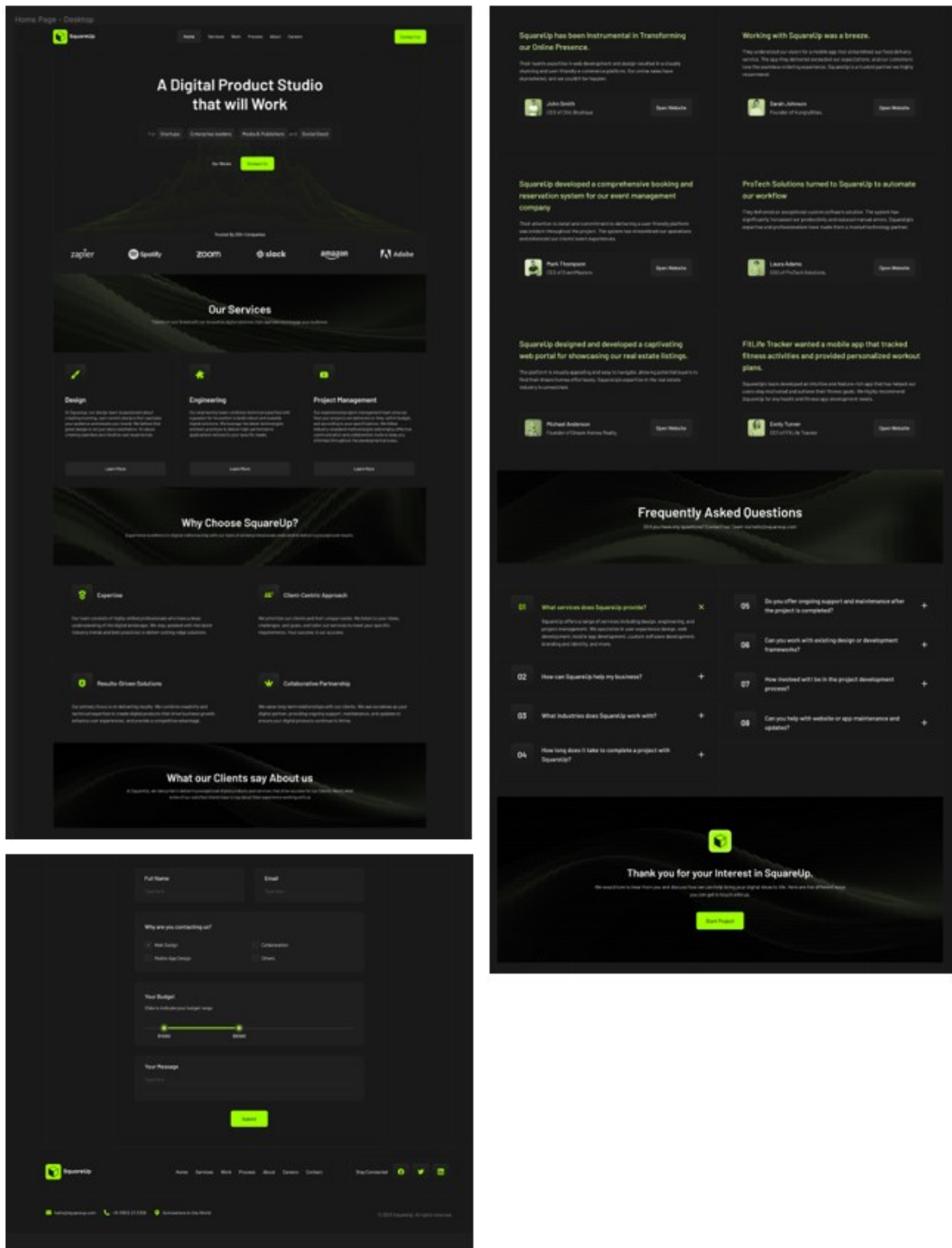


Figure 3. Tested company website design made by Praha.

Figma's plugin system was used to install the AI tools. React and TS were chosen as the preferred library and programming languages throughout the testing process due to their prevalence in the modern web development landscape. While the generated code wasn't directly integrated and deployed to a live website, each tool offered a built-in preview to visualize the rendered website within the Brave browser.

4 Results

This chapter presents the results of evaluating five AI design-to-code tools: Anima, Builder.io, Clapy, Locofy, and TeleportHQ. The evaluation focused on conversion accuracy, user experience, supported technologies, and conversion times.

4.1 Anima

Anima stood out for its fast conversion times, producing results in a short timeframe. However, its capabilities were limited by two key limitations: responsiveness and code accessibility. The generated websites exhibited poor responsiveness, meaning they didn't adapt effectively to different screen sizes. This resulted in layout problems, with elements potentially overlapping or displaying incorrectly on mobile devices. Although Anima's free version delivered semi-accurate results for both designs with functional links and smooth transitions, it lacked a code preview. The generated code wasn't accessible for the free version since it was behind a paywalled zip file. Additionally, the free version restricts users to a single project with only five pages. The results can be seen in Figure 4.

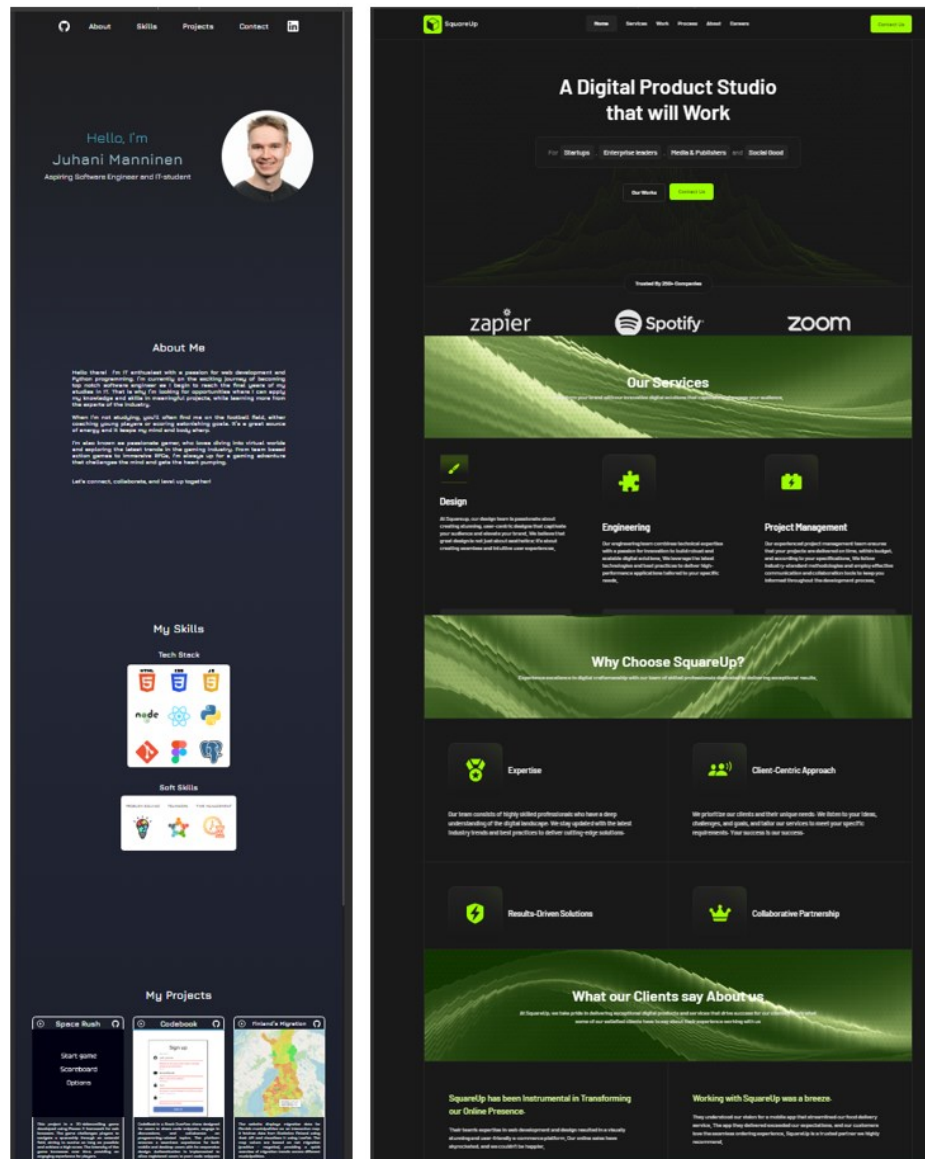


Figure 4. Anima's results.

4.1 Builder.io

Builder.io delivered accurate results, although there were some issues regarding to price slider, stretched or altered images, and misaligned navbar elements. The tool demonstrated fast conversion times, but it could only convert one design page at a time. In addition, it provides a user-friendly editor with drag-and-drop functionality. One notable drawback is the absence of a live preview feature, requiring users to view outputs within the editor or deploy them to a server for live previews. That is why interactive components such as buttons and text fields couldn't be tested. The results are showcased in Figure 5.



Figure 5. Builder. Io's results.

4.1 Clapy

Clapy, an entirely free open-source program exhibited mixed results. While it crashed Figma twice during the portfolio design conversion, it produced a webpage from the company website design (Figure 6). However, responsiveness was a major concern. The page skewed left even on desktop screens, and the footer appeared within another element at the top of the page. Notably, some dark and green images were unintentionally colored which deviated from the original design.

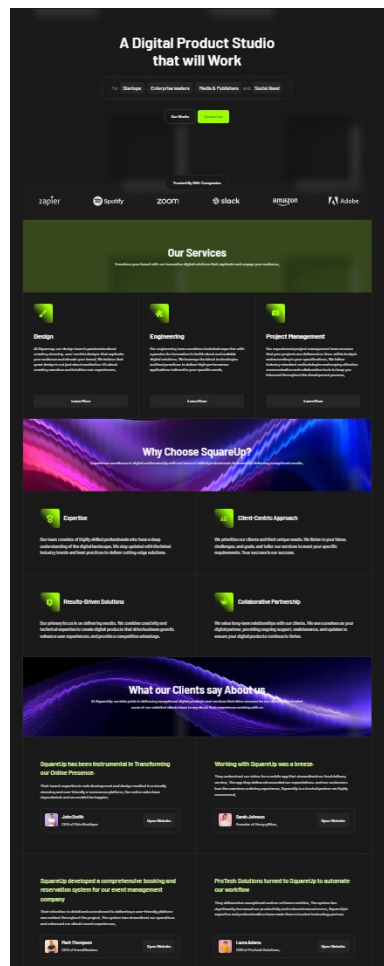


Figure 6. Clapy's results.

4.2 Locofy

Locofy offers accurate code conversion from Figma designs. However, it comes with limitations. While it supports many frameworks and other settings, a maximum of five frames can be converted at a time. This restriction becomes cumbersome when handling complex multi-page designs, which forces a segmented workflow. Additionally, performance issues were encountered during the company website conversion, with the tool freezing during the layer naming step. Additionally, responsiveness was poor, causing crucial information to be hidden on mobile view and some elements like images were too large, filling the screen. The portfolio design conversion faced further issues: links weren't added automatically, adding them using Locofy's UI broke the website, email links became text fields, and animated Figma components were converted to static images.

Despite these limitations, Locofy is a solid design-to-code tool due to its accuracy as depicted in Figure 7. Furthermore, Locofy was impressed with its user-friendly post-conversion editor. This intuitive UI allows developers with basic front-end knowledge to adjust the code directly.

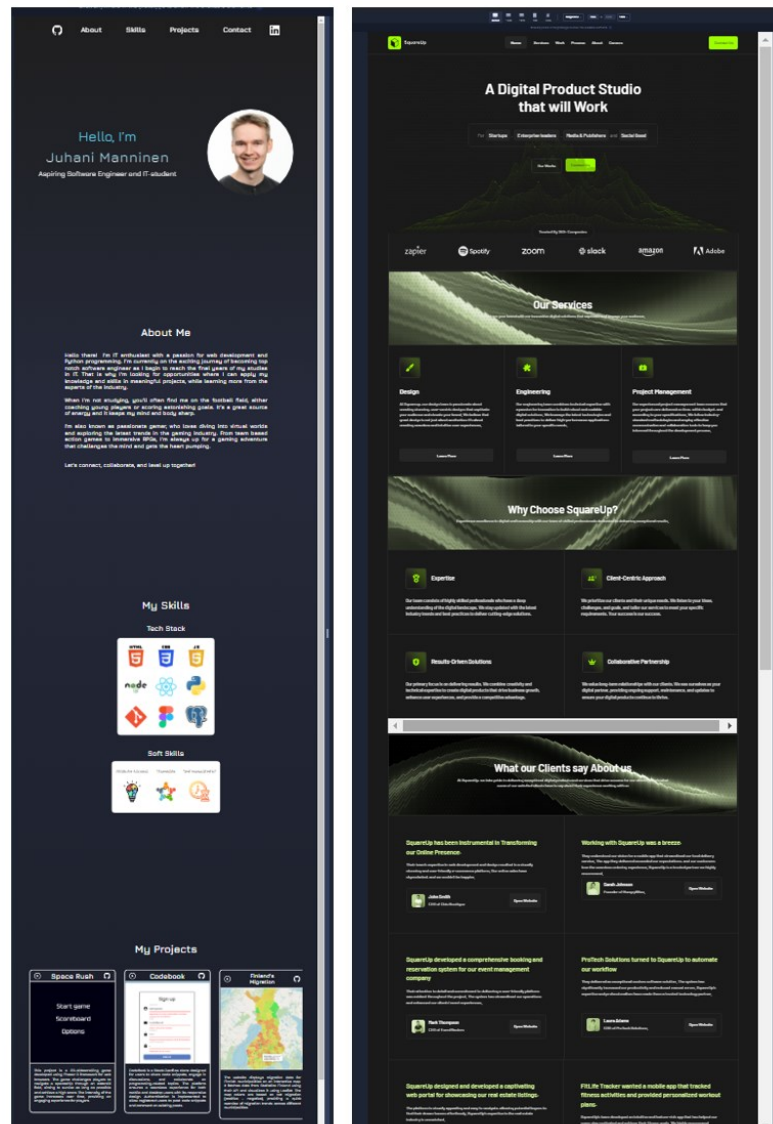


Figure 7. Locoify's results.

4.1 TeleportHQ

TeleportHQ managed to produce accurate results, but it also encountered numerous problems across both designs as shown in Figure 8. The about me text was missing from the portfolio page, responsiveness was poor in both designs, links, and animations weren't implemented, and half the company design images were excluded. The largest drawback was its performance. It crashed the browser while exploring export options and repeatedly

4.1 Summary of the AI Tools

Table 1 summarizes the test results and Table 2 summarizes the key features of the tested tools.

	Anima	Builder.io	Clapy	Locofy	TeleportHQ
Test results	+ Accurate results - Not responsive - Limited features for the free version	+ Accurate results - Not responsive	- Unstable - Low performance - Better alternatives	+ Accurate results - Not responsive	+ Accurate results - Not responsive - Unstable
Code conversion time	Company website (7pages): 1 minute	Company website (Home desktop page): Under 1 minute	Company website (Home desktop page): 3 minutes	Company website (5 desktop pages): 5 minutes Portfolio: 2 minutes	Company website (7 desktop pages): 6 min Portfolio: 1 minute
Maximum pages converted at the same time	5 (free version)	1	1	5	No page limit, but conversion can take a long time

Table 1. Summary of the test results.

	Anima	Builder.io	Clapy	Locofy	TeleportHQ
Supported technologies	Adobe XD CSS Figma HTML JS React Sketch Tailwind TS Vue	Adobe XD Angular Emotion Figma HTML JS Marko Mitos Qwik React React Native Solid Styled components Styled JSX Tailwind TS	Angular CSS CSS Modules Figma HTML React	Adobe XD Ant Design Bootstrap Chakra Figma Gatsby HTML/CSS JS Material UI Next.js React React Native TS (assuming this refers to TypeScript) Vuetify Vue	Angular Figma GitLab GitHub HTML JS Next.js Node.js Nuxt.js React Native Vercel Vue
Extensive documentation and guides	Yes	Yes	Yes	Yes	Yes
Pricing	0 – 150\$ / month	0 - +39\$ / month	Free	Free (Beta)	0 - +9\$ / month

Table 2. Features of the tools.

5 Discussion

This chapter discusses the gathered results during the evaluation of design-to-code AI tools. The discussion explores the tools' capabilities and limitations, highlighting areas for improvement and their suitability for different user groups.

5.1 Key Findings and Challenges

The evaluation of design-to-code tools revealed several crucial areas for consideration. A consistent challenge across all tools was their inability to convert designs into websites that adapt to different screen sizes. This highlights the need for further development in AI capabilities to ensure responsiveness in different screen sizes. Additionally, all tools exhibited limitations in recognizing and converting interactive elements like links and text fields. There is still much to be done to improve AI algorithms so they can identify design elements and understand their intended purpose.

Among the tested tools, Locofy demonstrated promising results in terms of accuracy and features. Anima and Builder.io achieved a similar level of accuracy, but they had some restrictions that Locofy didn't have. Clapy and TeleportHQ had significant performance, stability, and responsiveness issues, which is why they are not recommended until major updates are made to them.

5.2 Design-to-Code Tool Use Cases

Tools that performed well in the tests will increase the productivity of front-end developers by creating a solid foundation for further development. Tools like Locofy can streamline the workflow by converting basic design elements into code, allowing developers to focus on more complex logic and functionalities. However, for intricate features like customized animations or multi-page applications with complex user interactions, developers will likely find manual programming a better approach to achieve the desired level of control.

For users with no prior experience in design or programming, these tools may not be suitable due to the technical aspects involved. Website builders could offer a more user-friendly alternative, providing pre-built templates and drag-and-drop functionalities that eliminate the need for coding knowledge.

5.1 Aligning Study Results with Existing Literature

The results of this study align with the findings of Adelola and Oladokun [7], who observed that while these AI tools excel at generating basic prototypes, they exhibit inconsistencies when translating more complex design elements into code. For instance, the tested tools struggled to accurately convert the company website's price slider. This aligns with their observation that “While AI models excel at generating standard components, handling intricate or highly customized design requirements may pose challenges” [7]. Potential reasons for these limitations could be the difficulty for AI algorithms to interpret highly customized design elements or the need for more comprehensive training data. It's important to remember that the focus of this research was to assess the tools' ability to generate production-ready code. While they may be valuable for initial prototyping stages, programmers are still necessary for handling more advanced designs, as stated by Adelola's and Oladokun's research.

6 Conclusions

This research aimed to answer two primary questions:

RQ1: What are the capabilities and limitations of the available design-to-code AI tools?

RQ2: How useful are these tools in web development?

Design-to-code tools were able to convert Figma designs with varying degrees of accuracy. Additionally, features like code conversion speed, price, and supported technologies offered some differentiation among the tools. Common limitations across the tools included challenges with responsive design, recognizing and converting interactive elements, advanced features like animations, and creating multi-page web applications. Based on the mentioned qualities, Locofy emerged as the most promising tested design-to-code tool. However, it is important to acknowledge that this conclusion is based on a single user evaluation and may not be universally applicable. While design-to-code tools can expedite front-end development's initial development phases, considerable limitations remain. Developers likely need to refine and implement functionalities for more complex features.

This research had limitations that could be addressed in future research. The scope was confined to a single user's evaluation focusing on the looks only, excluding the quality of the code. Further research could benefit from including user groups with diverse backgrounds in design, programming, and non-technical expertise to gain broader perspectives. Implementing quantitative methods for comparing the accuracy of the results would provide more comprehensive results by utilizing metrics and statistical analysis to provide a more objective evaluation. The view of professional front-end developers would also grant insights into how usable the code is in a production environment. Additionally, evaluating a wider range of tools would offer a more thorough assessment of available options and their potential. Despite these limitations, this research sheds light on the design-to-code technology and its potential to streamline web development workflows.

References

- [1] ‘Top 10 best front end frameworks in 2024’, Blog | Imaginary Cloud. Accessed: Feb. 20, 2024. [Online]. Available: <https://www.imaginarycloud.com/blog/best-frontend-frameworks/>
- [2] ‘Front-End Development’, GeeksforGeeks. Accessed: Feb. 29, 2024. [Online]. Available: <https://www.geeksforgeeks.org/front-end-development/>
- [3] ‘Locofy.ai - ship your products 10x faster — with low code’. Accessed: Mar. 08, 2024. [Online]. Available: <https://www.locofy.ai/>
- [4] ‘Convert any Design to Code, Turn your UI designs to HTML code’, teleporthq.io. Accessed: Mar. 08, 2024. [Online]. Available: <https://teleporthq.io/>
- [5] ‘What is Artificial Intelligence and How Does AI Work? | Definition from TechTarget’, Enterprise AI. Accessed: Feb. 20, 2024. [Online]. Available: <https://www.techtarget.com/searchenterpriseai/definition/AI-Artificial-Intelligence>
- [6] B. Asiroglu *et al.*, ‘Automatic HTML Code Generation from Mock-Up Images Using Machine Learning Techniques’, in *2019 Scientific Meeting on Electrical-Electronics & Biomedical Engineering and Computer Science (EBBT)*, Istanbul, Turkey: IEEE, Apr. 2019, pp. 1–4. doi: 10.1109/EBBT.2019.8741736.
- [7] N. A. Ikumapayi, ‘Automated Front-End Code Generation Using OpenAI: Empowering Web Development Efficiency’, *SSRN Electron. J.*, 2023, doi: 10.2139/ssrn.4590704.
- [8] M. Bajammal, D. Mazinianian, and A. Mesbah, ‘Generating reusable web components from mockups’, in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, Montpellier France: ACM, Sep. 2018, pp. 601–611. doi: 10.1145/3238147.3238194.
- [9] A. Frick, *From Design to Code: A Study on Generating Production Code From User Interface Design Software*. 2022. Accessed: Feb. 19, 2024. [Online]. Available: <https://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-197681>
- [10] R. K. Yin, *Case Study Research: Design and Methods*. SAGE, 2009.
- [11] P. Runeson and M. Höst, ‘Guidelines for conducting and reporting case study research in software engineering’, *Empir. Softw. Eng.*, vol. 14, no. 2, pp. 131–164, Apr. 2009, doi: 10.1007/s10664-008-9102-8.
- [12] ‘Introduction – Locofy Docs’. Accessed: Mar. 06, 2024. [Online]. Available: <https://www.locofy.ai/docs/>